Presidential Speech Analysis

Tzu Feng Leung

```
# rm(list=ls())

library(rvest)
library(magrittr)
library(stringr)
library(assertthat)
library(testthat)
```

Problem 1

Sub Problem 1a

(a) From the website, you need to get the HTML file for each of the speeches. You'll need to start with this HTML file and extract the individual URLs for each speech. Then use that information to read each speech into R.

(b) For each speech, extract the body of the speech and the year of the speech.

(c) Convert the text so that all text that was not spoken by the president is stripped out and saved separately. For the Laughter and Applause cases, count the number of times each occurred in the speech.

(d) Extract the words and sentences from each speech as character vectors, one element per sentence and one element per word. Note that there are probably some special cases here. Try to deal with as much as you can, but we're not expecting perfection.

(e) For each speech count the number of words and characters and compute the average word length.

(f) Count the following words or word stems: I, we, America{,n}, democra{cy,tic}, republic, Democrat{,ic}, Republican, free{,dom}, war, God [not including God bless], God Bless, {Jesus, Christ, Christian}, and any others that you think would be interesting.

(g) The result of all of this activity should be well-structured data object(s) containing the information about the speeches.

(h) Make some basic plots that show how the variables have changed over time and (for presidents since Franklin Roosevelt in 1932) whether they seem to differ between Republican and Democratic presidents (the Republican presidents have been {Eisenhower, Nixon, Ford, Reagan, G.H.W. Bush, G.W. Bush, Trump} and the Democrats have been {Roosevelt, Truman, Kennedy, Johnson, Carter, Clinton, Obama, Biden}. Your response here does not have to be extensive but should illustrate what you would do if you were to proceed on to do extensive exploratory data analysis.

To accomplish the data scraping, I create modular functions to handle little tasks.

This function take in the table URL and extract all speech URLs from it.

```
getSpeechURLs <- function(URL) {
  # load in the html
  html <- read_html(URL)

  ## get all the a nodes from the table of [@class='table-responsive']
  listOfANodes <- html %>% html_elements(
                          xpath = "//table[@class='table-responsive']//td//a[@href]")
  # get all links from href
  speechURLs <- listOfANodes %>% html_attr('href')
  # get all links that constains 'www.presidency.ucsb.edu'
  speechURLs <- speechURLs[str_detect(speechURLs, pattern = 'www.presidency.ucsb.edu')]
  # keep only the unique links
  speechURLs <- unique(speechURLs)
  return(speechURLs)
}
```

This function extract all text from the speechURL and a vector containing all the paragraphs contained within the

tags in the html.

```
extractBodyOfSpeech <- function(speechURL) {
    if (class(speechURL)[1] == "xml_document") {
      html <- speechURL
    }
    else{
      html <- read_html(speechURL)
    }
    # extract body of speech
    listOfPNodes <- html %>% html_elements(xpath = "//div[@class='field-docs-content']//p")
    # extract the text out of <p>
    return( listOfPNodes %>% html_text() )
}
```

From the speechURL, extract the year.

```
extractYear <- function(speechURL) {
    if (class(speechURL)[1] == "xml_document") {
      html <- speechURL
    }
    else{
      html <- read_html(speechURL)
    }
    date <- html %>% html_elements(xpath = "//span[@class='date-display-single']") %>% html_text()
    date <- as.list(strsplit(date, split = ' ')[[1]])
    year <- date[3]
    return(unlist(year))
}
```

From the speechURL, extract the name of speaker.

```
extractPresidentName <- function(speechURL) {
    # <div class="field-title">
    if (class(speechURL)[1] == "xml_document") {
      html <- speechURL
    }
    else{
      html <- read_html(speechURL)
    }
    name <- html %>% html_elements(xpath = "//div[@class='field-title']//a") %>% html_text()
    return(name)
}
```

This function takes in a character vector containing all the sentences and return the number of times the pattern occur in the input.

```
getPatternCount <- function (listOfSentences, pattern){
    # for each sentence in the input list, count the occurrence of the pattern
    # Store the count in a list, return the sum of the list
    return( sum(sapply(listOfSentences, FUN = str_count, pattern)) )
}
```

From a list containing all paragraphs, divide each sentences up, and return the a character vector containing all the individual sentences.

```
## create a function to store each word in sentence in a character vector
## Usually, a sentence ends with '.', '!', or '?' followed with a space
## Make sure . is not after Mr, Ms, Mrs

getSentenceVecFromParagraphs <- function(bodyOfSpeech) {
  # input is character vector containing one or many paragraphs
  # output is list of all sentences
  # split each sentence up whenever we observe "!", ".", or "?" followed by a space
  return(unlist(str_split(bodyOfSpeech, pattern = "(?<!Mr|Ms|Mrs)(\\.|\\?|!)\\s+")))
}
```

From a list of sentences, extract each word; return a vector containing all the individual words.

```
getWordVecFromSentences <- function(sentenceVec) {
    assert_that(is.character(sentenceVec))
    ## takes a character vector that contains sentences
    ## str_extract_all returns a list of lists
    ## Inside the list of lists, the i^th list that contains all the words
    ##  from the i^th sentence
    ## finally use unlist() to convert the list of lists to just a simple character vector
    listOfLists <- str_extract_all(sentenceVec, pattern = "[a-zA-Z]+")
    return( unlist( listOfLists ) )
}
```

Take in the vector of words, return the total amount of characters.

```
getNumOfChar <- function(wordVec) {
    assert_that(is.character(wordVec))
```

```
    # takes charcter vector as input
    # for each word in wordVec, count the num of char of each word
    # for word count in a list
    # return sum of the list
    return (sum(sapply(wordVec, FUN = nchar)))
}
```

To detect the following interesting words, I used regular expression. Some common regex that I used are:

- (?i) make the search case insensitive
- \b(pattern)\b matches the pattern at the beginning or end of each word.

```
## (f) Count the following words or word stems: I, we, America{,n}, democra{cy,tic},
#   republic, Democrat{,ic}, Republican, free{,dom}, war,
#   God [not including God bless], God Bless, {Jesus, Christ, Christian},
#   and any others that you think would be interesting.
# patterns is a vector that conatins regular expressions that we wish to detect.
patterns <- c('\\bI\\b', '\\b(?i)we\\b', '(?i)america(n|ns)',
              '(?i)democra(cy|tic)', 'republic', 'Democra(t|tic)',
              '\\bRepublican(s)*\\b', '(?i)free(dom)*', '(?i)\\bwar(s)*\\b',
              '(?i)\\bgod(?!\\s+bless)\\b', '(?i)god\\s+bless', '(?i)(Jesus| Christ(ian)*)')

patternLabels <- c("I", "we", "America{,n}",
                   "democra{cy,tic}", "republic", "Democrat{,ic}",
                   "Republican", "free{,dom}", "war",
          "God [not including God bless]", "God Bless", "{Jesus, Christ, Christian}")
```

Takes in a sentence vector, a list of patterns, and the pattern labels. Returns the count for each pattern in a vector.

```
getListOfPatternCounts <- function(sentenceVec, patterns, patternLabels=NA){
  # create a empty vector
  patternCounts <- rep(NA, length(patterns))
  # for each pattern
  for (i in 1:length(patterns)) {
    # count the pattern and store it inside the vector
    patternCounts[i] <- getPatternCount(sentenceVec, pattern = patterns[i])
  }
  # give names to each entry of the vector
  names(patternCounts) <- patternLabels
  # returns the vector
  return(patternCounts)
}
```

This function takes in a character vector, a list of unwanted patterns. Strip out all unwanted patterns from input body of text and return the stripped body of text.

```
stripUnwantedPatterns <- function(bodyOfSpeech, patterns){
  for (pattern in patterns) {
    # replace unwanted patterns with ''
    bodyOfSpeech <- str_replace_all(bodyOfSpeech, pattern, replacement = '')
  }
  return(bodyOfSpeech)
}
```

This is everything put together:

```
getSpeechData <- function(speechURL) {
  # input is the speech URL
  # read the html
  speechHTML <- read_html(speechURL)
  #
  bodyOfSpeech <- extractBodyOfSpeech(speechHTML)
  presidentName <- extractPresidentName(speechHTML)
  yearOfSpeech <- extractYear(speechHTML)
  #
  sentenceVec <- getSentenceVecFromParagraphs(bodyOfSpeech)
  wordVec <- getWordVecFromSentences(sentenceVec)
  #
  patterns <- c('\\bI\\b', '\\b(?i)we\\b', '(?i)america(n|ns)',
                '(?i)democra(cy|tic)', 'republic', 'Democra(t|tic)',
                '\\bRepublican(s)*\\b', '(?i)free(dom)*', '(?i)\\bwar(s)*\\b',
                '(?i)\\bgod(?!\\s+bless)\\b', '(?i)god\\s+bless',
                  '(?i)(Jesus| Christ(ian)*)')

  patternLabels <- c("I", "we", "America{,n}",
                     "democra{cy,tic}", "republic", "Democrat{,ic}",
                     "Republican", "free{,dom}", "war",
                     "God [not including God bless]", "God Bless",
                     "{Jesus, Christ, Christian}")
  #
  patternCounts <- getListOfPatternCounts(sentenceVec, patterns, patternLabels)

  myPatterns <- c('(?i)peace', '(?i)thank' , '(?i)money', '\\b(?i)us\\b')
  myPatternLabels <- c('peace', 'thank', 'money', 'us')
  #
  numLaughter <- getPatternCount(sentenceVec, pattern = "\\[Laughter\\]")
  numApplause <- getPatternCount(sentenceVec, pattern = "\\[Applause\\]")
  #
  # strip out all "[]" with words in it
  unwantedPatterns <- c('\\[[a-zA-Z]+\\]')
  bodyOfSpeech <- stripUnwantedPatterns(bodyOfSpeech, unwantedPatterns)
  #
  numOfWords <- length(wordVec)
  numOfChar <- getNumOfChar(wordVec)
  avgWordLength <- numOfChar / numOfWords
  #
  myPatternCounts <- getListOfPatternCounts(sentenceVec, myPatterns, myPatternLabels)
  # in the end, return a list with data in it
  return(result = list(name = presidentName,
                       year = yearOfSpeech,
                       bodyOfSpeech = bodyOfSpeech,
                       sentenceVec = sentenceVec,
                       wordVec = wordVec,
                       patternCounts = patternCounts,
                       myPatternCounts = myPatternCounts,
                       numOfWords = numOfWords,
                       numOfChar = numOfChar,
                       avgWordLength = avgWordLength,
```

```
                        numLaughter = numLaughter,
                        numApplause = numApplause
                        ))
}
```

We first extract all URLs from the webpage. For each URL, extract the data and store the data in a list. In the end, we get a list of lists, where each element is list containing data of one speech.

```
tableURL = 'https://www.presidency.ucsb.edu/documents/presidential-documents-archive-guidebook/annual-m

speechURLs <- getSpeechURLs(tableURL)

## for each URL, apply the function getSpeechData to it
speechList <- lapply(speechURLs, getSpeechData)

names(speechList) <- c("name" ,"year" ,"bodyOfSpeech" ,"sentenceVec" ,"wordVec",
                        "patternCounts" ,"myPatternCounts" ,"numOfWords" ,"numOfChar",
                        "avgWordLength" ,"numLaughter" ,"numApplause" )
```

In our speech list, each element is a list conatining all the data we extract from the speech. One example is shown below.

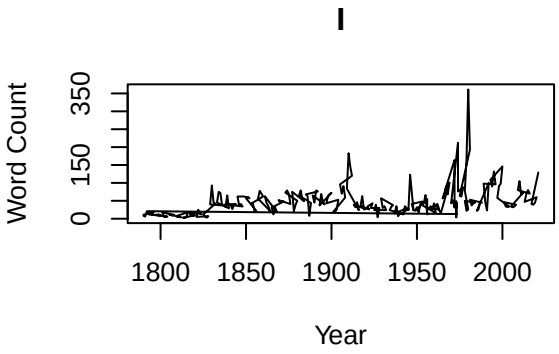
```
sapply(speechList[[1]], head)
```

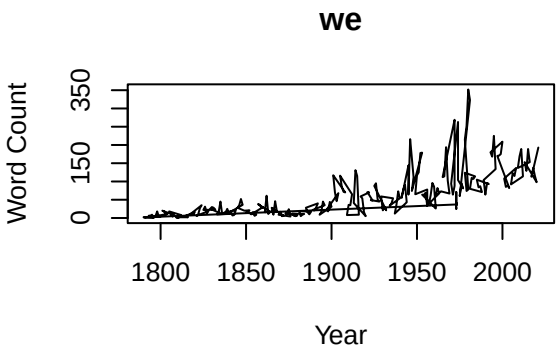
```
## $name
## [1] "Joseph R. Biden"
##
## $year
## [1] "2021"
##
## $bodyOfSpeech
## [1] "The President. Thank you. Thank you. Thank you. Good to be back. As Mitch and Chuck will unders
## [2] "Madam Speaker, Madam Vice President-no President has ever said those words from this podium. No
## [3] "First Lady-I'm her husband; Second Gentleman; Chief Justice; Members of the United States Congr
## [4] "Throughout our history, Presidents have come to this Chamber to speak to Congress, to the Natio
## [5] "I stand here tonight, 1 day shy of the hundredth day of my administration, hundred days since I
## [6] "Now, after just 100 days, I can report to the Nation: America is on the move again, turning per
##
## $sentenceVec
## [1] "The President"
## [2] "Thank you"
## [3] "Thank you"
## [4] "Thank you"
## [5] "Good to be back"
## [6] "As Mitch and Chuck will understand, it's good to be almost home, down the hall"
##
## $wordVec
## [1] "The"       "President" "Thank"     "you"       "Thank"     "you"
##
## $patternCounts
##                I             we     America{,n} democra{cy,tic}        republic
##              129            193              83              19               0
```

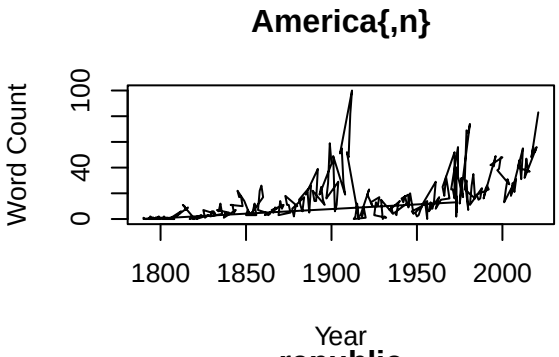
```
##   Democrat{,ic}
##              10
##
## $myPatternCounts
## peace thank money    us
##     1    10     3    18
##
## $numOfWords
## [1] 8211
##
## $numOfChar
## [1] 36036
##
## $avgWordLength
## [1] 4.388747
##
## $numLaughter
## [1] 3
##
## $numApplause
## [1] 1
```

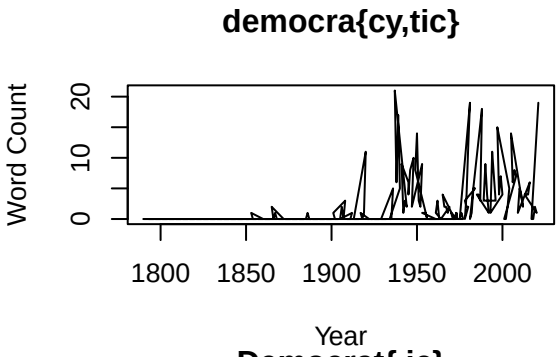
I use sapply to access element within speechList. Then use a for loop to plot the frequency of each word with respect to year.

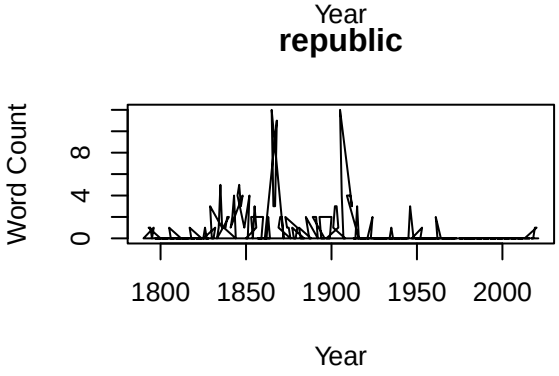
```
##
## from the list of lists, extract the interesting word counts
interestingWordCounts <- sapply(speechList, '[[' , 6)
## transpose it and store as data frame
interestingWordCounts <- as.data.frame(t(as.matrix(interestingWordCounts)))

## by the same logic, get all the years
years <- sapply(speechList, '[[' , 2)

## for each intersting word, make a scatter plot with respect to years
par(mfrow=c(2,2))
for (column in 1:ncol(interestingWordCounts)) {
  plot(years, interestingWordCounts[,column],
       main = colnames(interestingWordCounts[column]),
       ylab = "Word Count",
       xlab = 'Year',
       type = 'l')
}
```

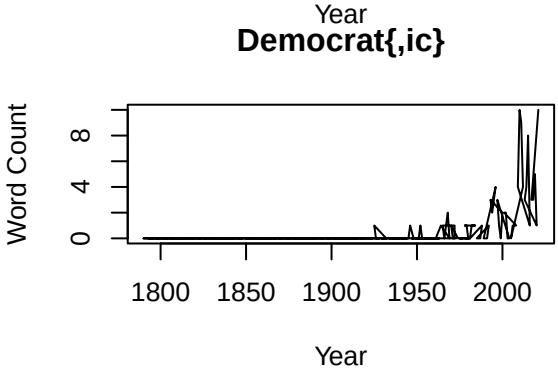

I
Word Count
0
150
350
1800
1850
1900
1950
2000
Year

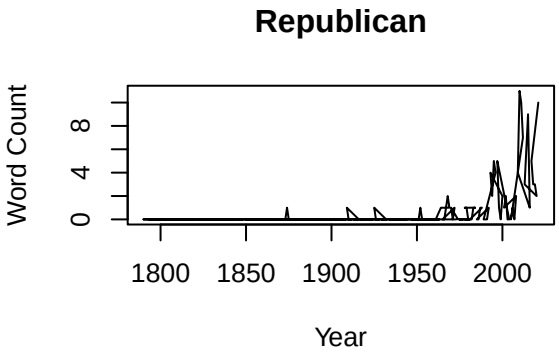

we
Word Count
0
150
350
1800
1850
1900
1950
2000
Year

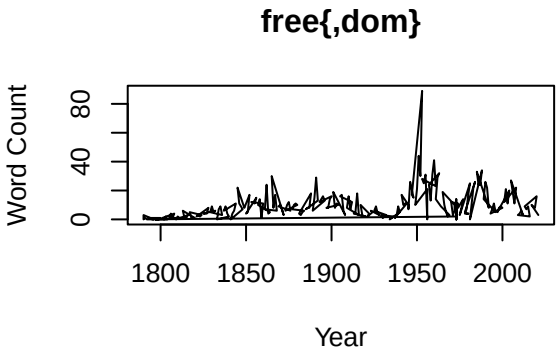

America{,n}
Word Count
0
40
100
1800
1850
1900
1950
2000
Year

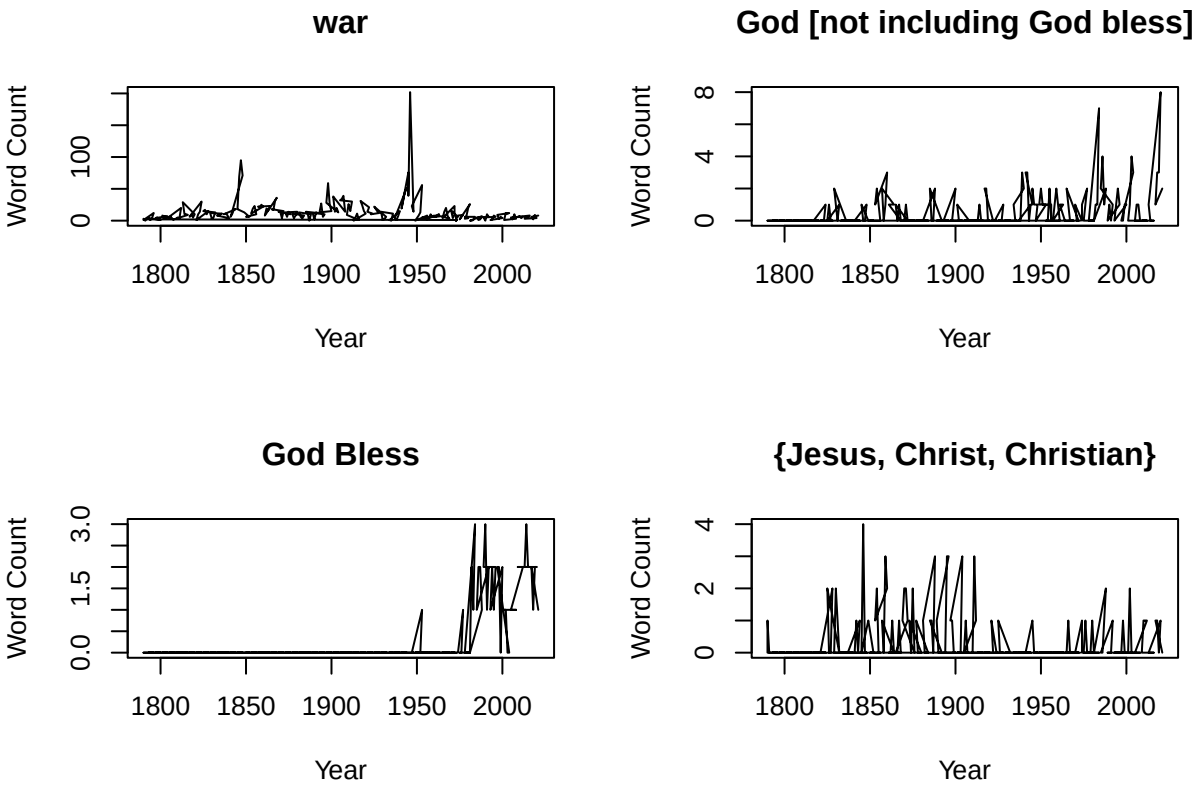

democra{cy,tic}
Word Count
0
10
20
1800
1850
1900
1950
2000
Year


republic
Word Count
0
4
8
1800
1850
1900
1950
2000
Year


Democrat{,ic}
Word Count
0
4
8
1800
1850
1900
1950
2000
Year


Republican
Word Count
0
4
8
1800
1850
1900
1950
2000
Year


free{,dom}
Word Count
0
40
80
1800
1850
1900
1950
2000
Year

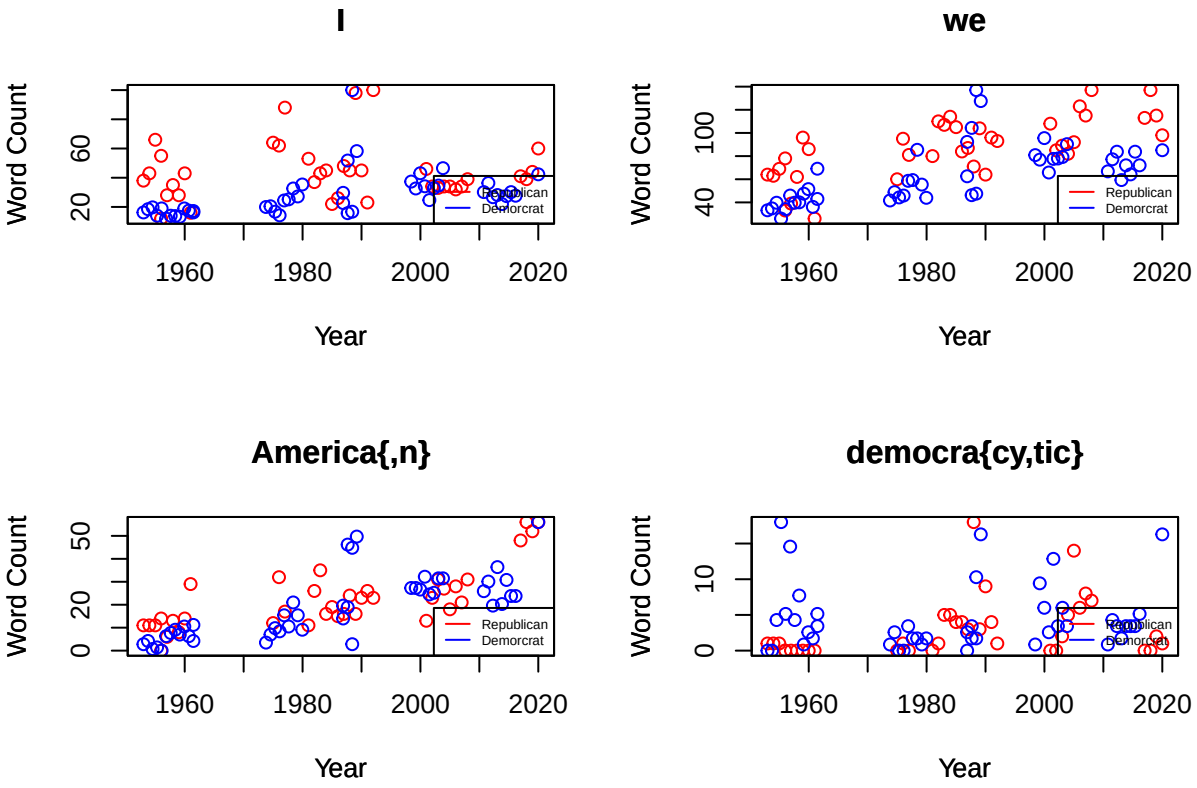


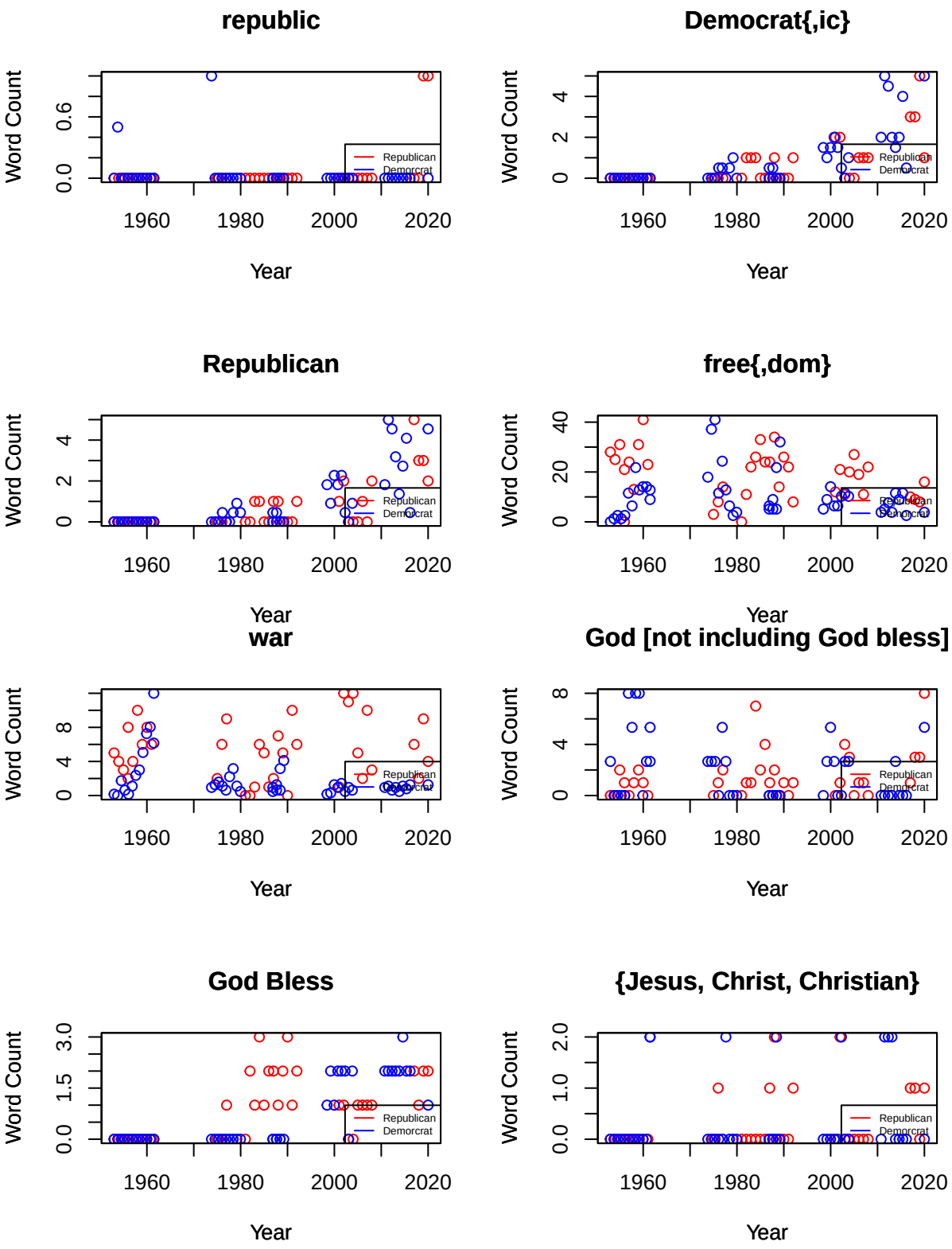

```
par(mfrow=c(1,1))
```

To compare how the variable differs for Republican and Democratic presidents. I extracted index of Republican and Democratic presidents, then make a scatter plot color coded by the party. Republican presidents are red and Democratic presidents are blue. Plots are shown below.

```
## vector containing all speaker names
speakerNames <- sapply(speechList, '[[' , 1)

## the names of Republican and Democratic presidents
republicans <- c("Donald J. Trump", "George W. Bush", "George Bush",
                 "Ronald Reagan", "Gerald R. Ford", "Richard M. Nixon",
                 "Dwight D. Eisenhower")
democrats <-c("Joseph R. Biden",  "Barack Obama", "William J. Clinton",
              "Jimmy Carter",   "Lyndon B. Johnson", "John F. Kennedy",
              "Harry S Truman", "Franklin D. Roosevelt")

## collect all the index of the republican party presidents
republicanIndex <- which(speakerNames %in% republicans)
## collect all the index of the democratic party presidents
democratIndex <- which(speakerNames %in% democrats)

par(mfrow=c(2,2))

## make plot for the republican party presidents
for (column in 1:ncol(interestingWordCounts)) {
  plot(years[republicanIndex], interestingWordCounts[,column][republicanIndex],
       main = colnames(interestingWordCounts[column]),
```

```
        ylab = "Word Count",
        xlab = 'Year',
        col = 'red'
        )
  par(new=TRUE)
  ## On the same plot make plot for the democratic party presidents
  plot(years[democratIndex], interestingWordCounts[,column][democratIndex],
     main = colnames(interestingWordCounts[column]),
     ylab = "Word Count",
     xlab = 'Year',
     col = 'blue',
     yaxt="n",
     xaxt="n"
     )
  legend("bottomright", legend=c("Republican", "Demorcrat"),
        col=c("red", "blue"), lty=1, cex = 0.5)
}
```

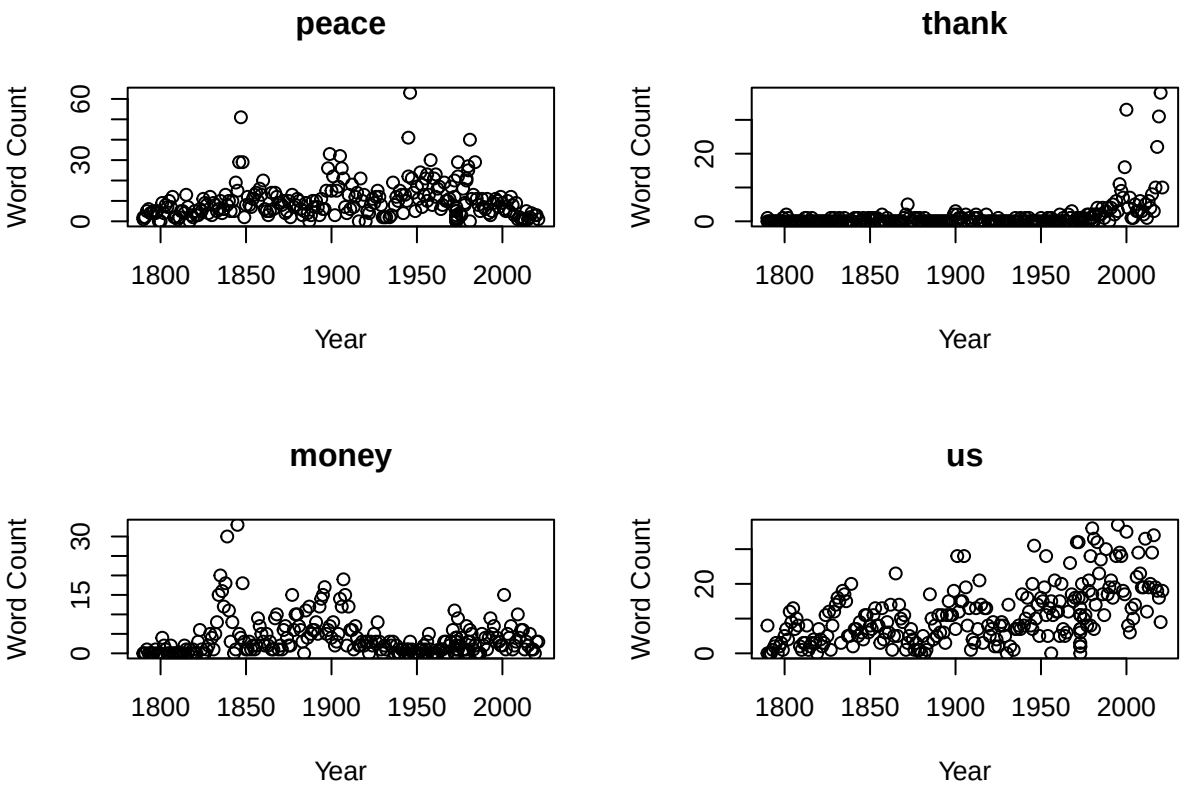




Use the same logic, I plotted words that I am interested in. Plots are below:

```
## below are the codes I wrote for
myWordCounts <- sapply(speechList, '[[' , 7)
myWordCounts <- as.data.frame(t(as.matrix(myWordCounts)))
```

```
par(mfrow=c(2,2))
for (column in 1:ncol(myWordCounts)) {
  plot(years, myWordCounts[,column],
       main = colnames(myWordCounts[column]),
       ylab = "Word Count",
       xlab = 'Year')
}
```

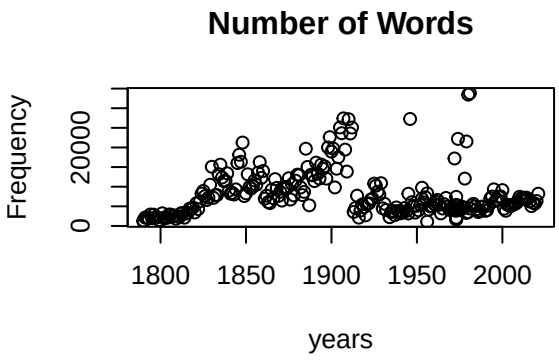


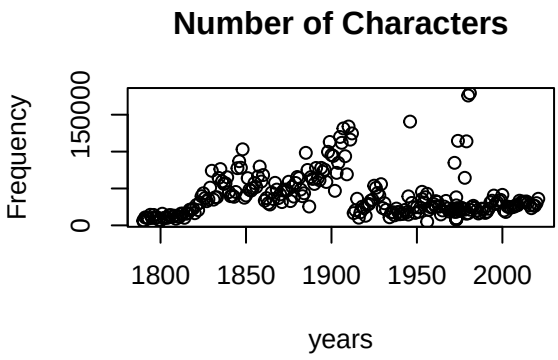

```
par(mfrow=c(1,1))
```

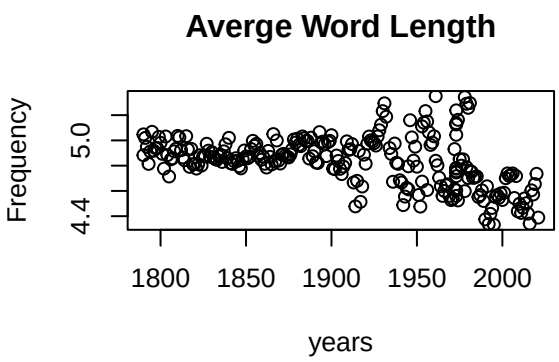
More plots:

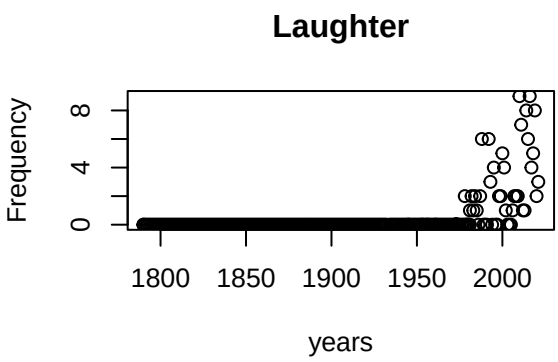
```
par(mfrow=c(2,2))

numOfWordsVec <- sapply(speechList, "[[", 8)
plot(years, numOfWordsVec,
     ylab = 'Frequency',
     main = 'Number of Words')
#
numOfCharVec <- sapply(speechList, "[[", 9)
plot(years, numOfCharVec,
     ylab = 'Frequency',
     main = 'Number of Characters')
#
avgWordLengthVec <- sapply(speechList, "[[", 10)
plot(years, avgWordLengthVec,
```

```
     ylab = 'Frequency',
     main = 'Averge Word Length')
#
numLaughterVec <- sapply(speechList, "[[", 11)
plot(years, numLaughterVec,
     ylab = 'Frequency',
     main = 'Laughter')
```










```
#
numApplauseVec <- sapply(speechList, "[[", 12)
plot(years, numApplauseVec,
     ylab = 'Frequency',
     main = 'Applause')
```